# GameSnake

Sebastián Cortés, Luis Carlos Díaz

February 2019

## 1 Hardware

El System On Chip sintetizado consta de un LM32 con 5 periféricos.

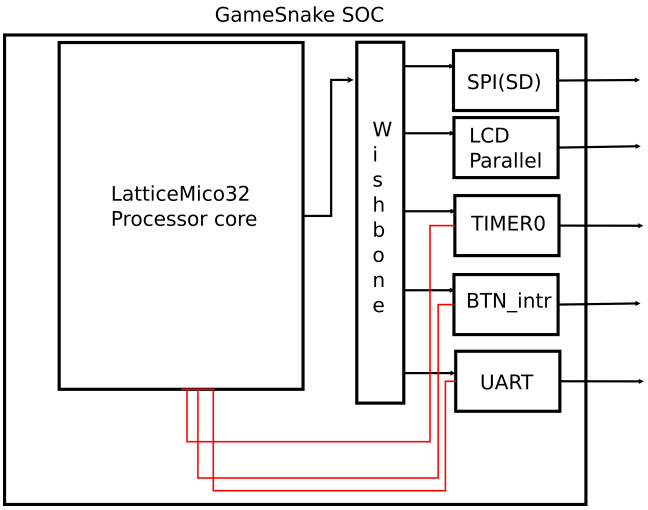



Figure 1: Modelo del sistema en chip.

| Periferico | Dirección |
|---|---|
| buttons | 0xe0004800 |
| ctrl | 0xe0000000 |
| lcd_test | 0xe0004000 |
| spi_sd | 0xe0005000 |
| timer0 | 0xe0002800 |
| uart | 0xe0001800 |
| uart_phy | 0xe0001000 |
| identifier_mem | 0xe0002000 |

Table 1: Mapa de memoria del SOC.

## 1.1 LCD parallel

La pantalla LCD seleccionada fue la ILI9641, con shiel predispuesta para manejar con 8 bits en paralelo. La escritura de los registros del chip ILI9641 se realizan conforme al diagrama de tiempos **??**.

Los registros del módulo son los mostrados en la tabla 2. se deben diferencias dos tipos de escritura en la LCD: commando y argumento.

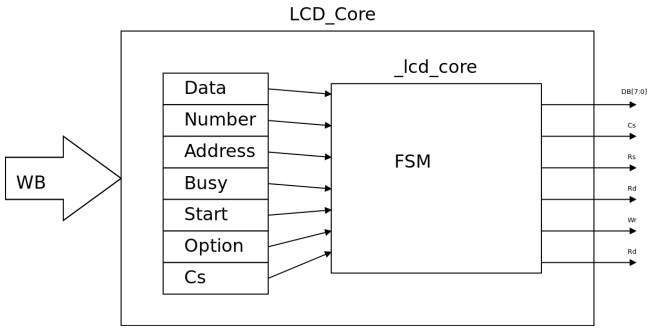



Figure 2: Diagrama del periférico.

ADDR es la dirección de cada registro del chip al que se le escriben comandos, DATA es la palabra o argumento que se va a escribir en dicha dirección, START configura el tipo de escritura, con OPTION se puede elegir el tipo de transmisión entre comando y argumento. CS es el selector de chip, en el caso se deja siempre en nivel bajo porque no se tienen más de un esclavo.

| Registro | Dirección | Tamaño | Tipo |
|---|---|---|---|
| lcd_test_DATA | 0xe0004000 | 8 | rw |
| lcd_test_ADDR | 0xe0004004 | 8 | rw |
| lcd_test_BUSY | 0xe0004008 | 1 | ro |
| lcd_test_START | 0xe000400c | 2 | rw |
| lcd_test_OPTION | 0xe0004010 | 1 | rw |
| lcd_test_CS_ | 0xe0004014 | 1 | rw |

Table 2: Mapa de memoria de lcd_core.

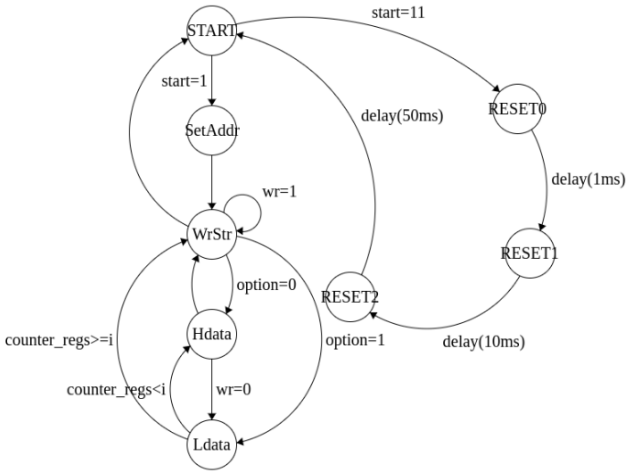



Figure 3: Máquina de estados propuesta.

Las declaraciones más importantes de cada estado de la máquina de la figura 1.1.3 son:

### 1.1.1 Start

Espera el valor del registro START para comenzar rutina de reset o de escritura.

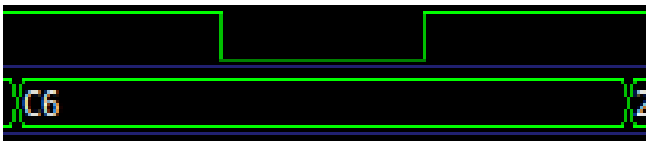
### 1.1.2 SetAddr

El número que hay en el registro ADDR se escribe en los pines de salida DB[7:0]

### 1.1.3 WRSTROBE

Es muy importante porque pone en nivel bajo y posteriormente en nivel alto el bit WR para hacer efectiva la escritura.

El tiempo que dura el bit WR en nivel bajo y en nivel alto es de 120ns. El dato válido en los pines DB[7:0] tiene una duración de 360ns.




Siempre se accede a este estado después de que se le escribe a los pines DB[7:0].

### 1.1.4 HData

Escribe en la salida DB[7:0] el byte más significativo del número (16bit) existente en el registro DATA.

### 1.1.5 LData

Escribe en la salida DB[7:0] el byte menos significativo del número (16bit) existente en el registro DATA.

### 1.1.6 Rutina Reset

En esta rutina se configura el pin RST de la tarjeta en nivel bajo, espera 1ms, luego en nivel alto, espera 10ms y luego lo vuelve a poner en nivel bajo y finalmente espera 50ms y termina.

En las figuras 4 y 6 se muestra el funcionamiento del modulo Lcd parallel en el analizador lógico

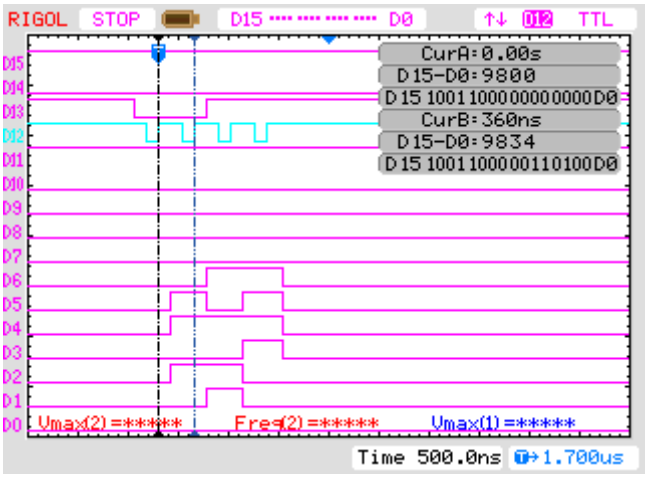



Figure 4: Escritura de datos.

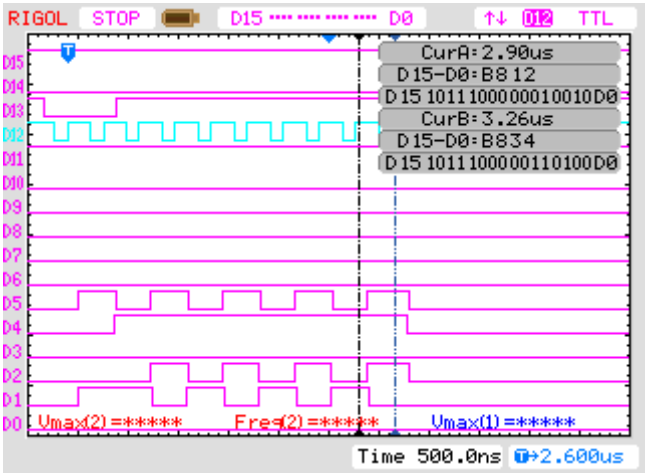



Figure 5: Escritura en la ram de gráficos

## 1.2 GPIO

Se utilizaron 5 botones para el juego. Los botones fueron manejados mediante interrupciones. Cada pulso genera un evento que es atendido por software para realizar el procedimiento requerido.

Litex cuenta con un periférico EventManager que controla las interrupciones. Se Realiza GPI, y se instancia un submódulo del tipo EventManager.

```
self.submodules.ev = EventManager()
self.ev.arriba = EventSourceProcess()
self.ev.abajo = EventSourceProcess()
self.ev.derecha = EventSourceProcess()
self.ev.izquierda = EventSourceProcess()


self.comb +=[
    self.ev.arriba.trigger.eq(signal[1]),
    self.ev.abajo.trigger.eq(signal [0]),
    self.ev.derecha.trigger.eq(signal[2]),
    self.ev.izquierda.trigger.eq(signal[3])
]
```

Figure 6: Escritura en la ram de gráficos

Se asigna el disparador como flanco de bajada para controlar las interrupciones y de esta manera se manejan los botones.

| Registro | Dirección | Tamaño | Tipo |
|---|---|---|---|
| buttons_in | 0xe0004800 | 1 | ro |
| buttons_dir | 0xe0004804 | 1 | ro |
| buttons_ev_status | 0xe0004808 | 1 | rw |
| buttons_ev_pending | 0xe000480c | 1 | rw |
| buttons_ev_enable | 0xe0004810 | 1 | rw |

Table 3: Mapa de memoria del GPIO

El registro $buttons_ev_pending$ es el que registra el evento de la interrupción, $buttons_ev_enable$

### 1.3 SPI SD

| Registro | Dirección | Tamaño | Tipo |
|---|---|---|---|
| spi_sd_config | 0xe0005000 | 32 | rw |
| spi_sd_xfer | 0xe0005004 | 32 | rw |
| spi_sd_start | 0xe0005008 | 1 | rw |
| spi_sd_active | 0xe000500c | 1 | ro |
| spi_sd_pending | 0xe0005010 | 1 | ro |
| spi_sd_mosi_data | 0xe0005014 | 32 | rw |
| spi_sd_miso_data | 0xe0005018 | 32 | ro |

Table 4: Mapa de memoria del SPI.

Los registros xfer y config configuran aspectos importantes de la comunicación como el tamaño de la palabra enviada, velocidad de transmisión, polaridad del chip selector. Tanto en $mosi_data$ como en $miso_data$

## 2 Software

### 2.1 LCD

Para escribir en los registros de la pantalla haciendo uso de los registros de control se usa la función *lcd_write_reg*.

```
static void lcd_write_reg(unsigned char address,unsigned short int value,
unsigned char option){
    lcd_test_ADDR_write(address);    //Escritura de la direccion
    lcd_test_DATA_write(value);      //Escritura del dato
    lcd_test_SENDONE_write(option); //Escritura de la direccion
    lcd_test_START_write(1);         //Inicio del ciclo de escritura
    lcd_test_START_write(0);
```

```
    while(lcd_test_BUSY_read());     //Esperar al fin del proceso
}
```

Haciendo uso de esta función se hace la secuencia de inicialización que consiste en escribir los registros mostrados a continuación.

| Reg | Data | Descripcion |
|---|---|---|
| 0x01 | 0x0000 | Software rest |
| 0x28 | 0x0000 | Display off |
| 0xc0 | 0x0023 | Power control 1 |
| 0xc1 | 0x0010 | Power control 2 |
| 0xc5 | 0x2b2b | Vcom Control |
| 0xc7 | 0x00c0 | Vcom Control 2 |
| 0x36 | 0x0088 | Memory access control |
| 0x3a | 0x0055 | Pixel format set |
| 0xb1 | 0x001b | Frame control |
| 0xb7 | 0x0007 | Entry mode set |
| 0x11 | 0x0000 | Sleep out |
| 0x29 | 0x0000 | Display on |

Table 5: Secuencia de inicialización.

Luego de la inicialización se procede a escribir en cada posición de la memoria de gráficos para esto se implementó la función *lcd_write_ntimes*, esta función ejecuta el comando 0x3e y envía el dato n veces.

```
static void lcd_write_ntimes(unsigned short int value,
unsigned int times){
    lcd_test_ADDR_write(0x3c);
    lcd_test_DATA_write(value);
    lcd_test_NUMBER_write(times);
    lcd_test_SENDONE_write(0);
    lcd_test_START_write(1);
    lcd_test_START_write(0);
    while(lcd_test_BUSY_read());
}
```

## 2.2 GPIO

Para leer los registros correspondientes a los botones se uso la función *buttons_in_read*, generada por Migen

```
static inline unsigned char buttons_in_read(void) {
        unsigned char r = csr_readl(0xe0006800);
        return r;
}
```

# References

[1] Ilitek, "TFT LCD Single Chip Driver", ili9341 datasheet, [Revisado Feb 2019]